Improving a 3-D Convolutional Neural Network Model Reinvented from VGG16 with Batch Normalization

Nontawat Pattanajak and Hossein Malekmohamadi
Institute of Artificial Intelligence
School of Computer Science and Informatics
De Montfort University, Leicester, UK
Email: {P17197105, hossein.malekmohamadi}@dmu.ac.uk

***Abstract*— It is challenging to build and train a Convolutional Neural Network model that can achieve a high accuracy rate for the first time. There are many variables to consider such as initial parameters, learning rate, and batch size. Unsuccessfully training a model is one of the most inevitable problems. In some cases, the model struggles to find a lower Loss Function value which results in a poor performance. Batch Normalization is considered as a remedy to overcome this problem. In this paper, two models reinvented from VGG16 are created with and without using Batch Normalization to evaluate their model performance. It is clear that the model using Batch Normalization provides a better result in terms of Loss Function value and model accuracy, which also achieves a very high accuracy rate. It also reaches the saturation point of the highest model accuracy faster than the model without Batch Normalization. This paper also finds that the accuracy of 3D Convolutional Neural Network model reinvented from VGG16 with Batch Normalization is at 91.2% which can beat many benchmarking results on UCF101 such as IDT [5], Two-Stream [10], and Dynamic Image Networks IDT [4]. The technique introduced in this paper shows a fast, reliable and accurate estimation of human activity type and could be used in smart environments.**

Keywords*— *Batch Normalization, Convolutional Neural Network, Deep Learning, and Human Activity Recognition

I. Introduction

Even though a human can understand contents in an image effortlessly, building a machine to understand images or videos is challenging. This is because a computer understands an image or a video as an array of numbers. Changing viewpoints, illumination, deformation, occlusion, and background clutter can change the values in the array of numbers [8]. Writing a program as a procedure is not a straightforward way to achieve this task. However, a Convolutional Neural Network (CNN) model can overcome this problem. CNN is currently one of the most frequently used Deep Learning (DL) techniques and plays a major role in computer vision. It can convolute a large number of image arrays to select only useful information, called feature extraction, for Neural Network to classify image features. When training a CNN model with a large dataset, it is challenging to build a model to achieve a high rate of accuracy for the first time. There are many parameters that affect the accuracy of training a CNN model. Selecting initial values for the optimizer, selecting a learning rate, or changing the weight values too much during training a CNN model can result in the difficulty finding the smallest error of Loss Function. When a Loss Function of a CNN model struggles to find a smaller error value, it leads to a low rate of model accuracy. Adding Batch Normalization (BN) is one of the most useful techniques that can be used to overcome these problems. This paper presents how to improve a CNN model built from scratch with Batch Normalization. The model is trained with UCF101 [12] which is a large-scale dataset of video classification - 101 classes of human activities.

II. Related Works

A. Batch Normalization

Batch Normalization is one of the most well-known techniques used to improve the training speed of DL models. It was introduced by Sergey Ioffe and Christian Szegedy in 2015 [19]. When training DL models especially using Stochastic Gradient Descent (SGD) as an optimizer, it requires careful consideration of a learning rate and initial values of the model parameters. Changing input data distribution, which cannot be avoided in a practical way, in each layer can also possibly cause "internal covariate shift" [19]. Each model layer needs to adapt itself continuously for the change of data distribution. If the distribution changes too much, a DL model will experience difficulty in finding a lower Loss Function value. As a result, the DL model will not be able to be trained successfully. Adding BN into DL models can prevent a large change in data distribution for each model layer [13]. Therefore, Batch Normalization is suggested for implementation to overcome this inevitable problem.

Batch Normalization was found as a method to reduce the internal covariate shift. The study was started by fixing the input distribution for each model layer when training is run. Whitening input data [19] can provide a fixed input distribution. It is an original concept that is used to reduce the internal covariate shift. However, whitening input data can be simplified by normalizing input data. In order to train a DL model with an SGD optimizer, Mini-batches are also introduced when normalizing input data. The algorithm of Batch Normalization proposed by Sergey Ioffe and Christian Szegedy is presented by Algorithm 1.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$
Parameters to be learned: γ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

Mini-batch mean: $\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$

Mini-batch variance: $\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2$

Normalize $\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$

Scale and shift: $y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$

Algorithm 1: Batch Normalization [19]

B. *UCF101 dataset*

UCF101 is one of the largest datasets in Human Activity Recognition (HAR) benchmarks. It was introduced by Khurram Soomro et al. in 2012 [12]. UCF101 contains 13,320 video clips which their lengths are between 1.06 and 71.04 seconds, the resolution is 320 × 240 pixel, and the frame rate is 25 *fps*. It consists of 101 activity classes such as Archery, Basketball, Diving, Playing Piano, and many others. The footage of this dataset is presented in Fig.1. When the dataset was first introduced, the performance baseline was 43.9% by Standard Bag of Words method using the implementation by Marcin Marszalek, et al. [16]. It implies that UCF101 is one of the most challenging datasets of HAR tasks. It consists of a large number of classes, video clips, and the contents of each video clips are varied.

Fig. 1. Footage of UCF101 benchmark [12]

C. *VGG16 – 2D CNN model*

VGG16 is a convolutional neural network presented by Karen Simonyan and Andrew Zisserman in 2015 [11]. The structure of VGG16 shown in Fig.2 consists of 13 convolutional layers which work to do feature extraction, and three fully connected neural network layers (FC) to classify input data. Maxpooling layers are also inserted at the end of each convolutional layer group. Moreover, Softmax layer is added at the end of the model to provide a classified result. VGG16 achieves a high rate of recognition performance at 89.3% of mean Average Precision (mAP) on VOC-2012 dataset which is an image dataset of action classification. The original VGG16, which is a 2D CNN model, supports only an image task. Therefore, VGG16 is reinvented to support HAR video dataset for this study.

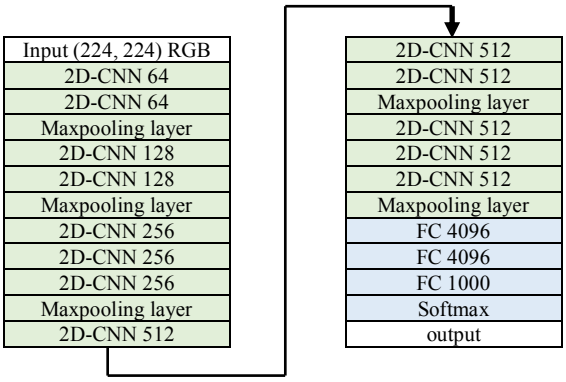

Fig. 2. The original VGG16 model [11]

III. EXPERIMENT

The process of this paper is divided into three sections such as doing video pre-processing, building 3D CNN models with and without Batch Normalization, and finally training both models to compare the accuracy of training and test data.

A. *Video Pre-Processing*

Even though originally the spatial dimensions of UCF101 are (320, 240, 3) which represent the width, height, and colour channel respectively for each image frame, they are compressed to (224, 224, 3) to support the models. In the temporal term, it is sampled equally 6 frames for each video clip. Therefore, the input data dimensions after being processed are (224, 224, 6, 3) which means width, height, temporal data, and channel respectively for each sample. In terms of label data, the dimension is (101) for each sample. This means that there are 101 classes of human activities. UCF101 contains a total of 13,320 samples. Therefore, the processed input data and label data dimensions are (13320, 224, 224, 6, 3) and (13320, 101) respectively. Thereafter, the processed input data and label data are stored and compressed to a single file as .npz data type to provide a convenient process for training and testing a DL model. The processed dataset (.npz file) size is 12.4 GB. Finally, the processed dataset is randomly separated into two groups for training and test data. The ratio between training and test data is 80:20. The Video Pre-Processing method is presented in Fig.3.

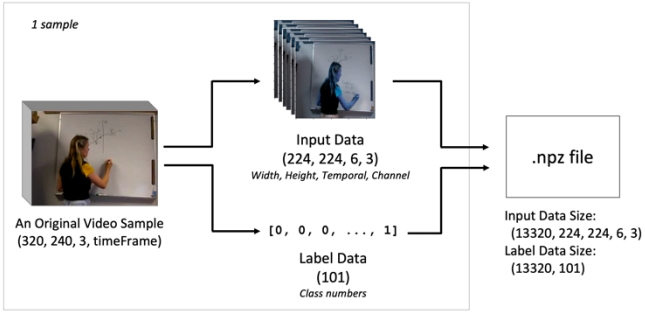

Fig. 3. Video Pre-Processing of UCF101 dataset

B. *Building 3D CNN models with and without BN*

There are two models used for this paper. They are built by Keras which is one of the most practical frameworks to build Deep Learning models.

Model 1 3D CNN **without BN**	Model 2 3D CNN **with BN**
3D-CNN 64	3D-CNN 64
Activation(relu)	Activation(relu)
3D-CNN 64	3D-CNN 64
Activation(relu)	Activation(relu)
3D Max Pooling	3D Max Pooling
	Batch Normalization
3D-CNN 128	3D-CNN 128
Activation(relu)	Activation(relu)
3D-CNN 128	3D-CNN 128
Activation(relu)	Activation(relu)
3D Max Pooling	3D Max Pooling
	Batch Normalization
3D-CNN 256	3D-CNN 256
Activation(relu)	Activation(relu)
3D-CNN 256	3D-CNN 256
Activation(relu)	Activation(relu)
3D-CNN 256	3D-CNN 256
Activation(relu)	Activation(relu)
3D Max Pooling	3D Max Pooling
	Batch Normalization
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D Max Pooling	3D Max Pooling
	Batch Normalization
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D-CNN 512	3D-CNN 512
Activation(relu)	Activation(relu)
3D Max Pooling	3D Max Pooling
	Batch Normalization
Flatten layer	Flatten layer
Neural Network 512 nodes with Activation(sigmoid)	Neural Network 512 nodes with Activation(sigmoid)
Softmax layer	Softmax layer

Fig. 4. 3D CNN models without BN, and with BN

Model 1 (3D CNN without BN): This model, which the structure for is shown in Fig. 4 (Model 1), consists of two sections, feature extraction and classification. The feature extraction is reinvented from VGG16 which is a 2D CNN model for image classification. To support video dataset, 2D Convolutional and 2D Max Pooling layers are replaced by 3D Convolutional and 3D Max Pooling layers respectively. For the classification section, there is one layer of neural network added. It consists of 512 nodes and uses Sigmoid as an activation function. Softmax is the last layer added at the end of the model to provide the results of the video classification.

Model 2 (3D CNN with BN): The second model shown in Fig.4 (Model 2) is similar to the first model except for adding Batch Normalization at the end of each Convolutional layer group. There are five convolutional groups such as 3D-CNN-64, 3D-CNN-128, 3D-CNN-256, and two 3D-CNN-512. This means that the input data is normalized before feeding into four convolutional groups (3D-CNN-128, 3D-CNN-256, and two 3D-CNN-512) and one neural network. The BN parameters used for this study are that momentum is 0.99 and epsilon is 0.001.

Adding BN increases the number of model parameters. Therefore, there are in total 44,450,085 and 44,455,973 parameters for the model without and with BN respectively.

C. *Training the models*

When training the models performed by 16 GB GPU (NVIDIA Tesla V100), Stochastic Gradient Descent is used to run as an optimizer. Two models are trained and tested in every epoch for 100 epochs with the processed data. The parameters for training both models are that batch size is 16, and the learning rate is 0.01.

IV. Result and Discussion

A 3D CNN model with and without Batch Normalization are compared. Three results are presented in this section such as the results of Loss Function values, model's accuracy, and confirming the accuracy of an improved model by a Cross-Validation method.

A. *The results of Loss Function values*

One of the most useful measurements of successfully training a CNN model is a low value of Loss Function which implies an error in the trained model output compared to the label data. Fig. 5 and 6 present the Loss Function values of training and test data respectively. It is clear that a model with BN provides a lower value than a model without BN at the starting point. The Loss Function value also drops dramatically when using BN while the value seems unchanged for the first five epochs when using no BN. Even though the Loss Function value of both models continues to decrease, a model without BN is not able to be trained after epoch 19. It cannot find a lower Loss Function value, as a result, the model has trained unsuccessfully. On the other hand, the model with BN has trained successfully and it reaches a Loss Function value at nearly zero, from approximately epoch 10 onwards for the training data.

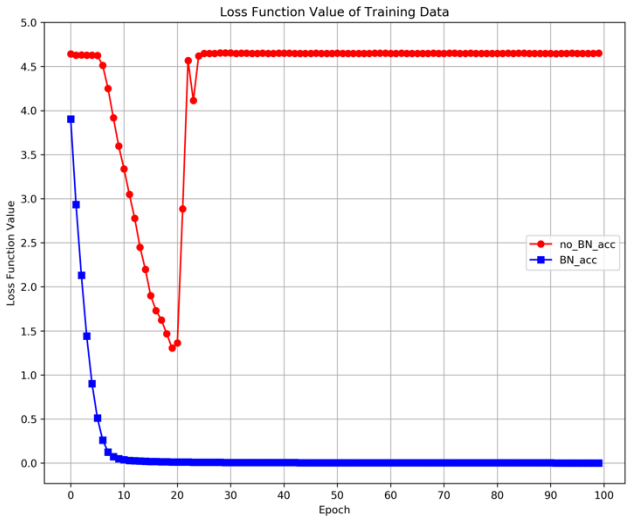

Fig. 5. Loss Function values of training data by epoch

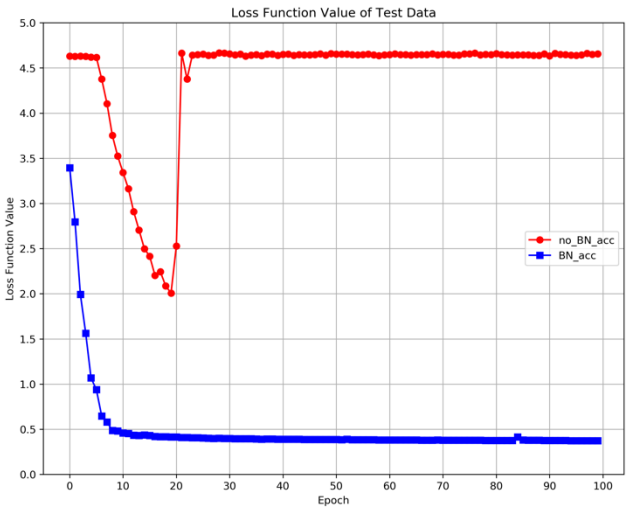

Fig. 6. Loss Function values of test data by epoch

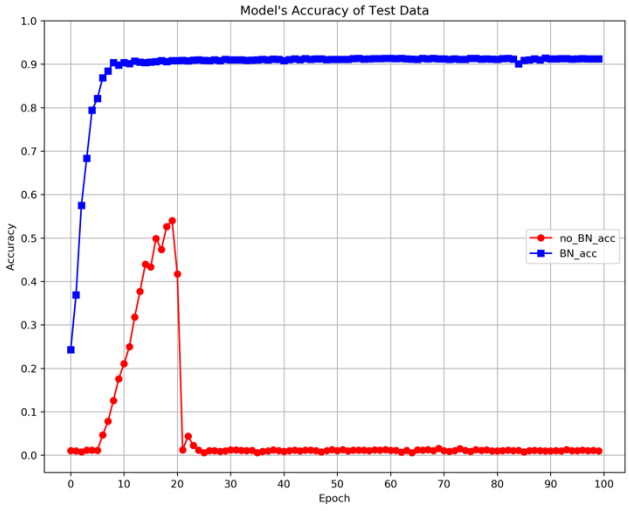

Fig. 8. Model's accuracy of test data by epochs

B. *Model's Accuracy Results*

The accuracy of both models is shown in Fig.7 and Fig.8 for training and test data respectively. The results show that a model with BN provides a higher rate of model accuracy than a model without BN. The accuracy of a model using BN increases dramatically and it reaches a saturation point at epoch 10 onward of approximately 100% and 90% for training and test data respectively.

On the other hand, even though the accuracy of a model without BN increases dramatically since epoch 8, it stops at epoch 19 which reaches its highest accuracy of 70.36% and 54.01% for training and test data. After that, the accuracy drops to nearly zero which means that this model is unreliable to be used for video classification of HAR tasks.

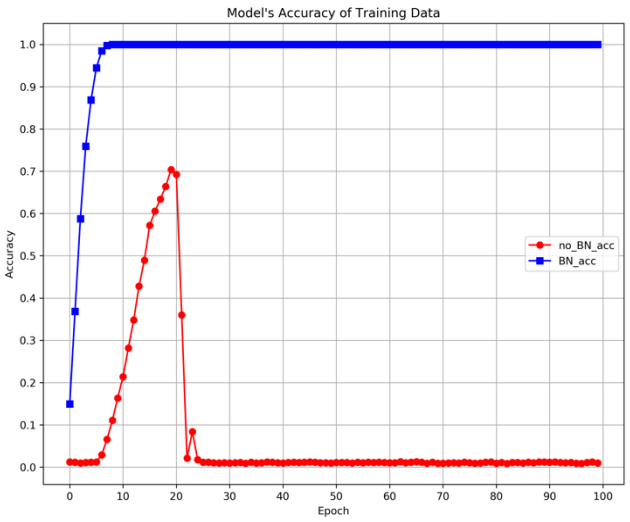

Fig. 7. Model's accuracy of training data by epochs

C. *Confirming the accuracy of the improved model by Cross-Validation method*

According to Fig. 7 and 8 which shows that the 3D CNN model reinvented from VGG16 with added BN, improved the model by producing a higher accuracy rate for both training and test data. However, training and testing a DL model on different data samples could provide different levels of accuracy even though doing it with the same dataset. Therefore, a Cross-Validation method [18] is performed to confirm model accuracy. For the Cross-Validation method, UCF101 dataset is duplicated to five groups. Each group is divided into two sections, training and test data in the ratio 80:20. Each group has a different sequence for training and test data. Then each data group is trained and tested on the improved model (the 3D CNN model with BN). The results of Cross-Validation are shown in Table 1. It found that the improved model can still achieve a high accuracy rate. The training and test's mean accuracy are 100% and 91.17% respectively.

TABLE I. CROSS-VALIDATION OF THE IMPROVED MODEL

CROSS-VALIDATION GROUP	ACCURACY (PERCENTAGE)	
	TRAINING DATA	TEST DATA
1	100	90.9534
2	100	90.4279
3	100	91.2162
4	100	92.5675
5	100	90.6906
AVG	100	91.1711

D. Comparing the improved model to state-of-the-art models

When comparing the accuracy of the improved model to state-of-the-art models which Joao Carreira and Andrew Zisserman study in 2018 [7] on a UCF101 benchmark, it found that the improved model can be ranked in a group of state-of-the-art accuracy as presented by Table 2.

TABLE II. STATE-OF-THE-ART MODELS ON THE UCF101 BENCHMARK

MODEL	ACCURACY (PERCENTAGE)
Two-Stream I3D, Imagenet+Kinetics pre-training [7]	98.0
ST-ResNet + IDT [2]	94.6
Temporal Segment Networks [14]	94.2
Two-Stream Fusion + IDT [1]	93.5
TDD + IDT [15]	91.5
3D CNN reinvented from VGG16 with BN (the work presented in this paper)	**91.2**
C3D ensemble + IDT, Sport 1M pre-training [3]	90.1
Dynamic Image Networks + IDT [4]	89.1
Two-Stream [10]	88.0
IDT [5]	86.4

E. Overall work discussion

There are two main points to present in this paper. Firstly, this paper suggests using BN to improve model accuracy when developing a DL model from scratch to make a DL model achieved a high accuracy rate. Two models are reinvented from VGG16. The difference between the models is one has the addition of BN. The model with BN provides a higher accuracy rate at the same epoch when comparing to the model without BN. Even though training both models for 100 epochs consumes the same amount of time – estimated at 13 hours, the speed of reaching the saturation point of the model with BN is also faster. In other words, it requires less epoch to train than the model without BN. The model with BN requires 10 epochs to reach the saturation point which consumes training time estimated at 1 hour and 20 minutes while the model without BN requires 19 epochs which consumes an estimated 2 hours and 30 minutes.

Secondly, 3D CNN model reinvented from VGG16 with BN is considered as an improved model. It consists of 44,455,973 parameters which require 178 MB memory (not include model structure) to store as a .hd5 data format. This model achieves a high accuracy rate of 91.2% which is within the state-of-the-art UCF101 benchmark results.

V. CONCLUSION

The results of training a CNN model with and without BN found that a model with BN provides a lower Loss Function value than another model for both training and test data. This results in the higher model accuracy rate. The model with BN achieves a model accuracy of approximately 100% and 90% for training and test data respectively, while a model without BN reaches its highest accuracy of 70.36% and 54.01% for training and test data. Once the accuracy of a model using BN reaches its highest point, it continues to remain at this level. However, the accuracy of a model without BN drops after it reaches its highest point. This study also observes that a model with BN requires nearly two times fewer epochs to reach its highest model accuracy, compared to a model without BN. Therefore, adding BN is a suggestion to provide a high accuracy rate of a CNN model especially when building from scratch.

In addition to these, it also finds that the improved model (3D CNN reinvented from VGG16 with BN) can achieve a high model accuracy of 91.17% after confirmation by the Cross-Validation method. This level of accuracy is in the range of state-of-the-art UCF101 benchmark results. However, the accuracy of training data is higher than test data, this means that the model is still confronting an overfitting problem [6]. There is room for model accuracy improvement in the future, by adding L1 and L2 regularization [9], adding dropout [17], or using more data to train the model.

REFERENCES

[1] C. Feichtenhofer, A. Pinz, A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in IEEE International Conference on Computer Vision and Pattern Recognition CVPR, Las Vegas, NV, USA, 2016, pp.1933-1941.

[2] C. Feichtenhofer, A. Pinz, R. P. Wildes, "Spatiotemporal residual networks for video action recognition," arXiv.org [Online], November 7 2016. Available: https://arxiv.org/abs/1611.02155

[3] D. Tran, et al., "Learning spatiotemporal features with 3d convolutional networks," in IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp.4489–4497.

[4] H. Bilen, et al., "Dynamic image networks for action recognition," in IEEE International Conference on Computer Vision and Pattern Recognition CVPR, Las Vegas, NV, USA, 2016, pp.3034-3042.

[5] H. Wang, C. Schmid, "Action recognition with improved trajectories," in IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 2013, pp.3551-3558.

[6] I. Bilbao, J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks," in Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 2007, pp.173-177.

[7] J. Carreira, A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," arXiv.org [Online], February 12 2018. Available: https://arxiv.org/abs/1705.07750

[8] J. Johnson, "Lecture 2 Image Classification," Stanford University School of Engineering [Online], August 11 2017. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture2.pdf

[9] J. Kukačka, V. Golkov, and D. Cremers, "Regularization for Deep Learning: A Taxonomy," arXiv.org [Online], October 29 2017. Available: https://arxiv.org/abs/1710.10686

[10] K. Simonyan, A. Zisserman, "Two-stream convolutional networks for action recognition in videos," arXiv.org [Online], November 12 2014. Available: https://arxiv.org/abs/1406.2199

[11] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org [Online], April 10 2015. Available: https://arxiv.org/abs/1409.1556

[12] K. Soomro, A.R. Zamir, M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," University of Central Florida [Online], November 2012. Available:

https://www.crcv.ucf.edu/wp-content/uploads/2019/03/UCF101_CRCV-TR-12-01.pdf

[13] L. Chen, et al., "Why batch normalization works? a buckling perspective," in IEEE International Conference on Information and Automation (ICIA), Macau, China, 2017, pp.1184-1189.

[14] L. Wang, et al., "Temporal segment networks: towards good practices for deep action recognition," in European Conference on Computer Vision, Amsterdam, Netherlands, 2016, pp.20-30.

[15] L. Wang, Y. Qiao, X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 2015, pp.4305–4314.

[16] M. Marszalek, I. Laptev, C. Schmid, "Actions in context," in IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 2929-2936.

[17] N. Srivastava, et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research 15, pp.1929-1958, 2014

[18] P. Refaeilzadeh, L. Tang, H. Liu, "Cross-Validation," Arizona State University [Online]. Available: http://leitang.net/papers/ency-cross-validation.pdf

[19] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv.org [Online], May 2 2015. Available: https://arxiv.org/abs/1502.03167